Key agreement in peer-to-peer wireless networks

Mario Čagalj, Srdjan Čapkun and Jean-Pierre Hubaux
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL) , CH-1015 Lausanne, Switzerland
mario.cagalj@epfl.ch, capkun@ucla.edu, jean-pierre.hubaux@epfl.ch

Preliminary version
To appear in *Proceedings of the IEEE (Special Issue on Security and Cryptography)*

***Abstract*— We present a set of simple techniques for key establishment over a radio link in peer-to-peer networks. Our approach is based on the Diffie-Hellman key agreement protocol, which is known to be vulnerable to the "man-in-the-middle" attack if the two users involved in the protocol do not share any authenticated information about each other (e.g., public keys, certificates, passwords, shared keys, etc.) prior to the protocol execution. In this paper, we solve the problem by leveraging on the natural ability of users to authenticate each other by visual and verbal contact. We propose three techniques: the first is based on visual comparison of short strings, the second on distance bounding, and the third on integrity codes; in each case, the users do not need to enter any password or other data, nor do they need physical or infra-red connectivity between their devices. We base our analysis on a well-established methodology that leads us to a rigorous modularization and a thorough robustness proof of our proposal.**

***Index Terms*— Wireless networks, Security, Key agreement protocols (Diffie-Hellman protocol), Man In the Middle Attacks (MITM), Message authenticators**

I. Introduction

As the popularity of mobile systems such as PDAs, laptops, and mobile phones increases every day, users tend to rely more on them in a growing number of situations. In this paper, we focus on the frequent case in which two people get together (e.g., at a meeting, or in the street) and make use of their devices to communicate with each other, or at least to exchange their (electronic) business cards. Clearly, the communication between these devices must be properly secured.

Very often, the two users will want the security between their devices to be peer-to-peer, thus operating independently from any authority. In practice, this means that the mobile devices must run a protocol to authenticate each other and to protect the data they exchange (to ensure confidentiality and integrity); the latter operation typically requires setting up a symmetric shared key. This key can be used to secure both immediate communications and communications that take place afterwards (e.g., when users exchange email over the Internet).

It is a common belief that peer-to-peer security is more difficult to achieve than traditional security based on a central authority; moreover, wireless communication and mobility are considered to be at odds with security. Indeed, jamming or eavesdropping is easier on a wireless link than on a wired one, notably because such mischief can be perpetrated without physical access or contact; likewise, a mobile device is more vulnerable to impersonation and to denial-of-service attacks.

In contrast to this widespread belief, we think that physical presence is the best way to increase mutual trust and to exchange information in a secure way. Indeed, authentication is straightforward, as users can visually recognize each other (if they meet for the first time, they can be introduced to each other by a common friend whom they trust; or they can check each other's ID). In order to establish a shared key, they can make use of a location limited channel (e.g., physical contact or infrared [1], [2]) between their two devices. The man-in-the-middle attack is considered to be infeasible in these conditions.

Recently, researchers have proposed solutions that run exclusively on a radio link (hence they do not require a special channel such as physical contact or infrared), which increases usability. To compensate for the much higher vulnerability of radio channels, in some solutions users are required to type a password in both devices [3]; in other solutions, they simply have to compare strings of words (the longer the string, the higher the security) [3], [4], [5].

In this work, we build on our previous work [5] and propose three approaches to the problem of *user-friendly* key agreement (and mutual authentication) in settings where the users do not share any authenticated information in advance. The first approach belongs to the family of solutions requiring the users to compare strings of words, whereas the other two approaches are completely novel; they are based on radio-channel specific techniques, namely, *distance-bounding* and *integrity-codes (I-codes)*. In addition, we make the following contributions: (i) we design protocols that are provably secure in a realistic communication model, (ii) we apply a modular approach to designing and analyzing the protocols, thus paving the way to the design of *re-usable* (provably secure) message transfer (MT) authenticators, and (iii) we significantly increase user-friendliness.

The paper is organized as follows. In Section II we state the problem and formulate our assumptions. In Section III we present our protocols. In Section IV we provide a security analysis of our protocols. In Section V we survey the related work. Finally, we conclude the paper in Section VI.

II. Problem statement and assumptions

We consider the following problem. Two users, each equipped with a personal device capable of communicating over a radio link, get together and want to establish a shared key. Although they can visually recognize each other, we assume that they do not share any authenticated cryptographic information (e.g., public keys or a shared secret) prior to this meeting. In addition, the users can communicate only over a radio channel (no infrared or physical ports are available). The challenge is the following: *How can the users establish a shared key in a secure way?*

A. Threats against radio-based systems

The Diffie-Hellman (DH) key agreement protocol [6] seems to be appropriate for the problem (and the set of assumptions) at hand; the DH key agreement protocol is believed to be secure against a passive adversary[1] (e.g., eavesdropping on a wireless link). Let us briefly review how the DH key agreement protocol works. To agree on a shared key, two users, Alice (A) and Bob (B) proceed as follows. A picks a random secret exponent X_A, and calculates the DH public parameter g^{X_A}, where g is a generator of a group of large order. B does the same, that is, he calculates g^{X_B}. Finally, A and B exchange the public parameters g^{X_A} and g^{X_B} and calculate the shared DH key as $K = g^{X_A X_B} = (g^{X_A})^{X_B} = (g^{X}_{B})^{X_A}$.

It is well known that the basic version of the protocol is vulnerable to an active adversary who uses a *man-in-the-middle* (MITM) attack. At first glance, it may seem that mounting the MITM attack against wireless devices that communicate over a radio link and are located within the radio communication range of each other can be perpetrated only by a sophisticated attacker. But this is not the case, as we will now explain by a simple example in the framework of Internet protocols.

The Address Resolution Protocol (ARP) [8] is used by the Internet Protocol (IP) to map IP network addresses to the hardware addresses used by a data link protocol. An attacker can send spoofed ARP-replies to the victim, who will consequently send all its packets to the attacking machine. In an experiment we conducted, we were able to redirect the traffic between two “legal” machines through an attacking machine, despite the fact that the two legal machines were in radio-communication range of each other. In this way, the attacker could perpetrate the MITM attack (by altering the DH parameters). For this attack we used a collection of publicly available tools for network auditing and penetration testing, called *dsniff* [9].

Of course, ARP-spoofing is not the only way to mount a MITM attack against wireless devices. Examples of more involved MITM attacks against Bluetooth [10] equipped devices can be found in [11] and [12].

Hence, our goal is to devise mechanisms that prevent the attacker from modifying the DH parameters without being noticed.

[1]This is true if the Computational Diffie-Hellman problem [7] is intractable.

B. Assumptions

We assume the users to be equipped with a computationally constrained personal device (e.g., a PDA). Each device is equipped with a radio transceiver (e.g., IEEE 802.11 [13]). We also assume that each device has a human-friendly interface (i.e., a screen and a keyboard).

In this paper, we will present our solution over the multiplicative group $\mathbb{G}$ with the generator g. Here, we take $\mathbb{G}$ to be a subgroup of $\mathbb{Z}_p^*$ of the prime order q, where $\mathbb{Z}_p^*$ is the multiplicative group of non-zero integers modulo a large prime p. However, the whole treatment here applies to any group in which the Decisional Diffie-Hellman (DDH) problem is hard. These are all groups in which it is infeasible to distinguish between quadruples of the form (g, g^x, g^y, g^{xy}) and quadruples (g, g^x, g^y, g^z) where x, y, z are random exponents. Furthermore, we assume that p and a generator g of $\mathbb{Z}_p^*$, $(2 \leq g \leq p-2)$ are selected and published. All devices are preloaded with these values[2].

Concerning the adversarial model, we adopt the Dolev-Yao threat model [7]. Thus, we assume that the attacker Mallory (M) controls the radio communication channel; he can obtain any message transmitted over the radio channel. M can initiate a conversation with any other user. However, we assume M to be *computationally bounded*. We further assume that the two parties involved in the communication do trust each other; otherwise, little can be done (a corrupted party can always disclose any secret information received by another party). Whenever we speak of the security of a given protocol, we implicitly assume that the users involved in the protocol (e.g., their devices) are not compromised.

C. Commitment schemes

Commitment schemes are an important cryptographic building block that we will be using in our protocols. In this subsection, we provide only an informal treatment of commitment schemes. The semantics of a commitment scheme are the following: (i) a user who commits to a certain value cannot change this value afterwards (we say that the scheme is *binding*), (ii) the commitment is hidden from its receiver until the sender “opens” it (we say that the scheme is *hiding*).

A commitment scheme transforms a value m into a commitment/opening pair (c, d), where c reveals no information about m, but (c, d) together reveal m, and it is infeasible to find $\widehat{d}$ such that $(c, \widehat{d})$ reveals $\widehat{m} \neq m$. Now, if Alice wants to commit a value m to Bob, she first generates the commitment/opening pair $(c_A, d_A) \leftarrow \mathrm{commit}(m)$, and sends c_A to Bob. To open m, Alice simply sends d_A (and m if necessary) to Bob, who runs $\widehat{m} \leftarrow \mathrm{open}(\hat{c}_A, \hat{d}_A)$; we denote with $\hat{x}$ the message at the receiver's side when message x is sent over a public (unauthentic) channel. If the employed commitment scheme is “correct”, at the end of the protocol we must have $m = \widehat{m}$. In our security analysis, we assume the usage of an ideal commitment scheme. We are now ready to describe our protocols.

[2]We stress here that we could let users select and communicate to each other their own parameters p and g. However, this would come at the expense of the number (and size) of messages to be exchanged between the users, and our goal is to keep key exchange protocols as simple as possible.

$$\begin{array}{rcl}
\text{Alice} & & \text{Bob} \\
\text{Given } ID_A, g^{X_A} & & \text{Given } ID_B, g^{X_B} \\
\text{Pick } N_A \in_U \{0,1\}^k & & \text{Pick } N_B \in_U \{0,1\}^k \\
m_A \leftarrow 0\|ID_A\|g^{X_A}\|N_A & & m_B \leftarrow 1\|ID_B\|g^{X_B}\|N_B \\
(c_A, d_A) \leftarrow \mathrm{commit}(m_A) & & (c_B, d_B) \leftarrow \mathrm{commit}(m_B) \\
& \xrightarrow{\quad c_A \quad} & \\
& \xleftarrow{\quad c_B \quad} & \\
& \xrightarrow{\quad d_A \quad} & \widehat{m}_A \leftarrow \mathrm{open}(\hat{c}_A, \hat{d}_A) \\
\widehat{m}_B \leftarrow \mathrm{open}(\hat{c}_B, \hat{d}_B) & \xleftarrow{\quad d_B \quad} & \text{Verify } 0 \text{ in } \widehat{m}_A;\ i_B \leftarrow N_B \oplus \hat{N}_A \\
\text{Verify } 1 \text{ in } \widehat{m}_B;\ i_A \leftarrow N_A \oplus \hat{N}_B & & \\
\multicolumn{3}{c}{\text{If } i_A = i_B, \text{ Alice and Bob output "Accept" } \widehat{m}_B \text{ and } \widehat{m}_A, \text{ respectively.}}
\end{array}$$

Fig. 1. Operation of Diffie-Hellman key agreement protocol with String Comparison (DH-SC)

III. Proposed solutions

The common characteristic of the three protocols that we propose in this section is that they all aim at ensuring the integrity of DH public parameters (g^{X_A}, g^{X_B}), rather than the integrity of the agreed key K. An important advantage of such an approach is based on the following observation: people build trust in each other when they meet in person; secure communication is usually needed after their first physical meeting (typically over the Internet). Clearly, in such a scenario, it is not necessary to compute the shared DH key immediately. This "expensive" computation (a modular exponentiation) can be postponed to some later time, when (remote) secure communication is needed. As a consequence, if the solution to the problem of integrity checking of DH public parameters is not too computationally demanding, the process of integrity checking can be carried out on computationally constrained devices. This is very important since, when on the move, people are often equipped with only this kind of devices (e.g., mobile phones, PDAs).

A. Diffie-Hellman key agreement based on short String Comparison (DH-SC)

The simplest way to check the validity of the exchanged DH public parameters for Alice (A) and Bob (B), is to report the exchanged public parameters g^{X_A} and g^{X_B} to each other and then perform a comparison of them. The comparison of the exchanged values can be performed by looking at the screen of the communicating party, or by reading aloud the values to be compared. Although this approach provides very strong security, it is clearly impractical because it requires A and B to compare rather large streams of digits. A possible way to make visual (and verbal) verification easier for A and B is to represent the DH public parameters in a more readable form by, for example, significantly reducing the number of digits to be compared (and potentially encoding the bits in a more readable form as in RFC 2289 [14]). However, in this way, many different (long) DH public parameters translate to the same (short) bit string (the check value). This may give some advantage to a potential attacker.

Another simple approach consists in first exchanging g^{X_A} and g^{X_B} over a public channel, and in turn, verifying (for example, visually) that $h(g^{\hat{X}_A}\|g^{X_B})$ matches $h(g^{X_A}\|g^{\hat{X}_B})$, where h is a hash function satisfying certain security properties and "$\|$" denotes a concatenation. In order for this approach to be usable, the output of the hash function h should be truncated to a relatively short length (e.g., around 50 bits). With this approach, an adversary is successful if he can find values a and b such that $h(g^{X_A}\|a) = h(b\|g^{X_B})$; an adversary can find a *collision* on the truncated output of $h(\cdot)$. Note that it is not sufficient for an adversary to find any collision on $h(\cdot)$. On the contrary, the adversary is not constrained to finding a *second pre-image*[3] for a single fixed image value g^{X_A} or g^{X_B}; an adversary controls inputs to $h(\cdot)$ through the values a and b. Furthermore, the outcome of the used hash function is truncated (e.g., 50 bits long). Therefore, even if $h(\cdot)$ is a second pre-image resistant hash function, this still may not be a sufficient guarantee that the adversary cannot find a collision between truncated $h(g^{X_A}\|a)$ and $h(b\|g^{X_B})$. In Section V, we will describe a similar problem with an approach proposed by [15], where the users compare the truncated output of a hash function applied to the shared key $K = g^{X_A X_B}$.

In order to make the approach based on string comparison usable, it is essential to make a *proper trade-off between security and usability*. We propose a provably secure protocol called DH-SC (Diffie-Hellman key agreement with String Comparison) that achieves *optimal* trade-off between security and usability.

The protocol unfolds as shown on Fig. 1; we note here that this is an optimized version of the protocol that we proposed in [5] (some credit for this optimization goes to Serge Vaudenay). Both Alice (A) and Bob (B) have selected their secret exponents X_A and X_B, respectively, randomly from the set $\{1, 2, \ldots, q\}$ (q being the order of $\mathbb{G}$) and calculated DH public parameters g^{X_A} and g^{X_B}, respectively. A and B proceed by generating k-bit random strings N_A and N_B, respectively. Finally, A and B calculate commitment/opening pairs for the concatenations $0\|ID_A\|g^{X_A}\|N_A$ and $1\|ID_B\|g^{X_B}\|N_B$, respectively. Here, 0 and 1 are two public (and fixed) values that are used to prevent a *reflection attack* (Section IV-D). ID_A and ID_B are human readable identifiers belonging to parties A and B (e.g., e-mail addresses).

The following four messages are exchanged over a radio link. In the first message, A sends to B the commitment c_A. B responds with his own commitment c_B. In turn, A sends out d_A, by which A opens the commitment c_A. B checks the correctness of the commitment/opening pair $(\hat{c}_A, \hat{d}_A)$ and verifies that 0 appears at the beginning of $\hat{m}_A$. If the

[3]For a given x, x' is said to be a second pre-image if $x \neq x'$ and $h(x) = h(x')$ [7].

verification is successful, B sends, in the fourth message, d_B, by which B opens the commitment c_B. A in turn checks the commitment and verifies that 1 appears at the beginning of $\hat{m}_B$. If this verification is successful, A and B proceed to the final phase (Fig. 1).

In the final phase, A and B first generate the verification strings i_A and i_B, respectively, as shown on Fig. 1 ($\oplus$ is the bitwise "xor" operation). The length of each of these strings is k. Finally, Alice and Bob (as users) simply compare i_A and i_B. If they match, Alice and Bob accept each other's DH public parameters g^{X_A} and g^{X_B} and the corresponding identifiers ID_A and ID_B as being authentic. At this stage, Alice and Bob can safely generate the corresponding secret DH key ($g^{X_A X_B}$).

We assess the security of our protocol in Section IV. Here, we only state the result. To do this, we define formally what we mean by a secure protocol.

Definition 1: We say that any protocol $\Pi(k,(A,B))$ is a *secure protocol enabling authentication of DH public parameters* between A and B if the (polynomial-time) attacker M cannot succeed in deceiving A and B into accepting DH public parameters different than g^{X_A} and g^{X_B}, except with a satisfactorily small probability $\mathcal{O}(2^{-k})$ (i.e., a constant c in $\mathcal{O}(\cdot)$ is such that $c \ll 2^k$).

To state the result about the security of DH-SC protocol, we need two additional security parameters (k was already introduced before: it is the length of verification strings i_A and i_B). We denote with γ the maximum number of sessions (successful or abortive) of the DH-SC protocol that any party can participate in. We further assume that there are n parties that are using the DH-SC protocol. The following result is proven under the assumption that an ideal commitment scheme is used.

Theorem 1: The probability that an attacker succeeds against the DH-SC protocol is bounded by $n\gamma 2^{-k}$. Therefore, for the appropriately chosen parameter k, DH-SC is a secure protocol enabling authentication of DH public parameters.

Remark 1: The probability of success by the attacker as stated in Theorem 1, refers to the success against any one among all DH-SC protocol runs (successful or abortive); in other words, the attacker does not care which parties communication he breaks/influences. On the contrary, the probability that the attacker is successful against a fixed (targeted) party is only $\gamma 2^{-k}$.
The proof is given in Section IV. Let us give an example of possible values for the above parameters. Assume there are at most $n = 2^{20}$ parties using our protocol and each party can participate in at most $\gamma = 2^{20}$ sessions (successful or abortive) in its lifetime. Then, by choosing $k = 55$ we obtain that the highest probability of success by the attacker (having seen a huge number $n\gamma = 2^{40}$ of protocol runs) is at most $n\gamma 2^{-k} = 2^{-15}$. Note that k also represents the length of the verification strings i_A and i_B to be compared by users. To make this job easier for users, we can encode $k = 55$ bits into ℓ short words from some predefined dictionary (e.g., RFC 2289 [14]). For example, in order to have $\ell = 5$, where each word is 4 characters long, each user would have to store a dictionary of $2^{\frac{k}{\ell}} = 2^{11} = 2048$ 4-character words. It is clear that ℓ can be reduced further by using larger dictionaries.

B. Diffie-Hellman key agreement based on Distance Bounding (DH-DB)

In this section, we describe a key agreement protocol that is based on verifiable principal proximity, achieved through distance bounding. We call our protocol Diffie-Hellman with Distance-Bounding (DH-DB). The protocol ensures the secure establishment of a shared key between two parties A and B if there are no other parties that are closer to A or to B than they are to each other. In this section, we assume that the pair of devices have the means to accurately estimate the distance between themselves (later in this section we discuss the possible techniques for this purpose).

The proximity check between the two devices is performed through distance bounding [16]: each device upper-bounds its distance to the device with which it is agreeing on a key. The measured distance appears on both device displays. The users then visually check whether there are other users/devices closer to them than the displayed distance bounds. If this is not the case, the exchanged DH public parameters and the corresponding identities are accepted.

The DH-DB protocol is shown on Fig. 2. Note that the protocol on Fig. 2 is actually built upon the DH-SC protocol (Fig. 1). The only difference is that the verification of the authentication strings i_A and i_B (in the DH-DB protocol) is performed through Brands and Chaum's distance bounding protocol [16]. Thus, Alice (A) and Bob (B) exchange the commitment/opening pairs (c_A, d_A) and (c_B, d_B) in the first four messages in exactly the same way as in DH-SC protocol. Furthermore, A and B perform all necessary verifications as in the DH-SC protocol. Finally, A and B calculate k-bit verification strings i_A and i_B. As we can see on Fig. 2, A and B also exchange commitments $c_A^{'}$ and $c_B^{'}$ to concatenations $0\|R_A$ and $1\|R_B$; again, 0 and 1 serve to protect against the reflection attack.

Upon reception of the commitments $c_A^{'}$ and $c_B^{'}$, the devices execute distance bounding by exchanging bit by bit all the bits of R_A, R_B, i_A and i_B as shown on Fig. 2. During distance bounding, the devices measure round-trip times between sending a bit and receiving a response bit. The device estimates the distance-bound to the other device by multiplying the round trip time by the speed of light in the case of the radio or by the speed of sound in the case of ultrasound communication.

Having exchanged R_A, R_B, i_A and i_B, A and B open $c_A^{'}$ and $c_B^{'}$ by sending out $d_A^{'}$ and $d_B^{'}$, which they then use to retrieve $\hat{R}_B$ and $\hat{R}_A$, respectively. A and B then use $\hat{R}_B$ and $\hat{R}_A$ to retrieve $\hat{i}_B$ and $\hat{i}_A$; this is done by performing a series of k "xor" operations as shown on Fig. 2. Finally, A and B verify $\hat{i}_B$ and $\hat{i}_A$ against i_A and i_B; note that this verification is now done by the devices A and B, whereas in the DH-SC protocol this comparison is performed by users A and B.

Having successfully verified $\hat{i}_B$ against i_A and $\hat{i}_A$ against i_B, the devices A and B display the measured distance bounds on their screens. The users A and B then visually verify that there are no other users/devices in their vicinity (in what we call the *integrity region* of A and B; see Fig. 3). If the displayed distance bound corresponds to the distance to the closest device, the users accept the exchanged DH public

Alice		Bob
Given ID_A, g^{X_A}		Given ID_B, g^{X_B}
Pick $N_A, R_A \in_U \{0,1\}^k$		Pick $N_B, R_B \in_U \{0,1\}^k$
$m_A \leftarrow 0 \Vert ID_A \Vert g^{X_A} \Vert N_A$		$m_B \leftarrow 1 \Vert ID_B \Vert g^{X_B} \Vert N_B$
$(c_A, d_A) \leftarrow \text{commit}(m_A)$		$(c_B, d_B) \leftarrow \text{commit}(m_B)$
$(c'_A, d'_A) \leftarrow \text{commit}(0 \Vert R_A)$		$(c'_B, d'_B) \leftarrow \text{commit}(1 \Vert R_B)$
	$\xrightarrow{c_A, c'_A}$	
	$\xleftarrow{c_B, c'_B}$	
	$\xrightarrow{d_A}$	$\widehat{m}_A \leftarrow \text{open}(\hat{c}_A, \hat{d}_A)$
$\widehat{m}_B \leftarrow \text{open}(\hat{c}_B, \hat{d}_B)$	$\xleftarrow{d_B}$	Verify 0 in $\widehat{m}_A$; $i_B \leftarrow N_B \oplus \hat{N}_A$
Verify 1 in $\widehat{m}_B$; $i_A \leftarrow N_A \oplus \hat{N}_B$		
	— distance-bounding phase —	
The bits of R_A are $R_{A1}, R_{A2}, \ldots, R_{Ak}$		The bits of R_B are $R_{B1}, R_{B2}, \ldots, R_{Bk}$
The bits of i_A are $i_{A1}, i_{A2}, \ldots, i_{Ak}$		The bits of i_B are $i_{B1}, i_{B2}, \ldots, i_{Bk}$
$\alpha_1 \leftarrow R_{A1} \oplus i_{A1}$	$\xrightarrow{\alpha_1}$	
	$\xleftarrow{\beta_1}$	$\beta_1 \leftarrow R_{B1} \oplus i_{B1} \oplus \hat{\alpha}_1$
	$\ldots$	
$\alpha_i \leftarrow R_{Ai} \oplus i_{Ai} \oplus \hat{\beta}_{i-1}$	$\xrightarrow{\alpha_i}$	Measure delay between β_{i-1} and $\hat{\alpha}_i$
Measure delay between $\hat{\beta}_i$ and α_i	$\xleftarrow{\beta_i}$	$\beta_i \leftarrow R_{Bi} \oplus i_{Bi} \oplus \hat{\alpha}_i$
	$\ldots$	
$\alpha_k \leftarrow R_{Ak} \oplus \hat{\beta}_{k-1}$	$\xrightarrow{\alpha_k}$	Measure delay between β_{k-1} and $\hat{\alpha}_k$
Measure delay between $\hat{\beta}_k$ and α_k	$\xleftarrow{\beta_k}$	$\beta_k \leftarrow R_{Bk} \oplus \hat{\alpha}_k$
	— end of distance-bounding phase —	
	$\xrightarrow{d'_A}$	$0 \Vert \hat{R}_A \leftarrow \text{open}(\hat{c}'_A, \hat{d}'_A)$
$1 \Vert \hat{R}_B \leftarrow \text{open}(\hat{c}'_B, \hat{d}'_B)$	$\xleftarrow{d'_B}$	
$\hat{i}_{Bi} \leftarrow \alpha_i \oplus \hat{\beta}_i \oplus \hat{R}_{Bi} \quad (i = 1, \ldots, k)$		$\hat{i}_{A1} \leftarrow \hat{\alpha}_1 \oplus \hat{R}_{A1}$
Verify $i_A \overset{?}{=} \hat{i}_B$		$\hat{i}_{Ai} \leftarrow \hat{\alpha}_i \oplus \beta_{i-1} \oplus \hat{R}_{Ai} \quad (i = 2, \ldots, k)$
		Verify $i_B \overset{?}{=} \hat{i}_A$
Alice and Bob visually verify if there are other users/devices in their vicinity (the "integrity region").		

Fig. 2. Operation of the Diffie-Hellman key agreement with Distance Bounding.

parameters g^{X_A} and g^{X_B} and the corresponding identities ID_A and ID_B as being authentic; otherwise, they reject them. This last step is important as it guarantees that the exchanged messages in the protocol preserved their integrity, meaning that they cannot have been maliciously modified or generated by an adversary, but only by the closest party.

1) Properties of DH-DB protocol: In DH-DB, the MITM attack is prevented by the proximity verification. We define the *integrity region* of users A and B as the union of two spheres each centered at the position of devices A and B with radii equal to the distance d between devices A and B (see Fig. 3). If the users can visually verify that there are no other users/devices within the integrity region and if the distance-bounding phase is secure, then the integrity of messages i_A and i_B is respected; i.e., i_A and i_B sent from A and B will reach B and A, respectively, unchanged. Note here that the security of the distance-bounding phase relies on the fact that the attacker does not learn R_A and/or R_B until the end of this phase; all that M knows are commitments c'_A and c'_B. Therefore, R_A and R_B guarantee to A and B that the attacker cannot send the bits, in the distance-bounding phase, earlier than receiving the previous bit; for this reason, it cannot appear to be closer than it actually is.

If attacker M is not within the integrity region, he will not be able to send messages to A such that it seems that it is placed on the same (or shorter) distance from A as B. With this, the integrity of i_A and i_B is preserved as if users A and B exchanged i_A and i_B face to face (e.g., voice communication). Since i_A and i_B are actually authentication strings from the DH-SC protocol, by verifying that i_A and i_B match, users A and B are guaranteed that messages m_A and m_B are authentic, except with a satisfactorily small probability (see Theorem 1).

A nice property of this protocol is that it does not depend on the power ranges of the devices, but solely on their proximity d. Specifically, the closer the parties are, the smaller the integrity region is, and the harder it is for the adversary to get into the region without being noticed by the honest parties. For example, getting the devices as close as $d = 30$ cm should be a sufficient guarantee, even for the most demanding users, that no adversary (be it even a small device) resides in the

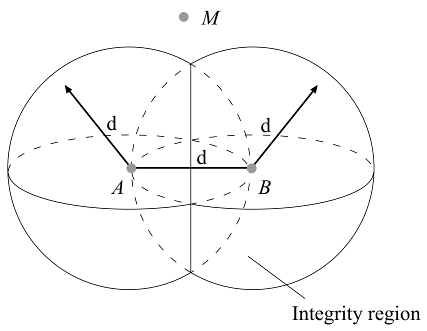

Fig. 3. Integrity region of users A and B (d is the distance between users' devices)

corresponding integrity region.

2) Implementation: We envision two possible implementations of DH-DB: with radio (RF) and with ultrasound (US). Both exhibit equal security guarantees, but require different equipment attached to the devices. We briefly report on how these implementations have been addressed so far. Brands and Chaum [16] propose a distance bounding protocol that can be used to verify the proximity of two devices connected by a radio link; it requires devices with a high (nanosecond) precision-of-time measurement. To the best of our knowledge, the only commercial technique that achieves such precision, and achieves therefore a high precision-of-distance measurement, is Ultra Wide Band (UWB). In [17], Fontana has demonstrated that with UWB, distances can be measured with an error margin of up to 15 cm.

Sastry, Shankar and Wagner [18] propose a distance bounding protocol based on ultrasound and radio wireless communication (a similar technique was also proposed by Waters and Felten [19]). Ultrasound-based distance bounding requires only millisecond time measurement precision, but of course it needs each device to be able to communicate via ultrasound. Ultrasound-based distance bounding has centimeter precision.

In both radio-frequency and ultra-sound solutions, the response time (the XOR operation and the reversion of the transceiver) of the challenged principal must be tightly bound and predictable.

C. Diffie-Hellman key agreement with Integrity Codes (DH-IC)

In this section, we propose a mechanism that reduces the involvement of the users to a single press of a button. This mechanism is based on what we term an *integrity code*.

1) Concept of Integrity codes (I-codes): I-codes are to be used with communication media (channels) for which we can ensure that it is not possible to block emitted signals without being detected, except with a negligible probability.

Definition 2: An integrity code is a seven-tuple $(\mathcal{S}, \mathcal{M}, \mathcal{E}, \mathcal{P}, l, t, e_c)$, where the following conditions are satisfied:

1) $\mathcal{S}$ is the (finite) set of possible source states[4]
2) $\mathcal{M}$ is the set of binary sequences with t 1's and $l-t$ 0's
3) $\mathcal{E}$ is the set of source encoding rules $e_s : \mathcal{S} \to \mathcal{M}$, where $e_s \in \mathcal{E}$ is an injective function (hence $|\mathcal{M}| \geq |\mathcal{S}|$)
4) $\mathcal{P}$ is the set consisting of two power levels 0 and p with $p > 0$, i.e., $\mathcal{P} \equiv \{0, p\}$
5) $e_c : \mathcal{M} \to \mathcal{P}^l$ is the channel modulation function satisfying the following rules: (i) the symbol "1" is transmitted using power level p; (ii) the symbol "0" is transmitted using power level 0.

Note from the above definition that the set $\mathcal{M}$ consists of l-bit long codewords, each containing a uniquely ordered sequence of t symbols "1" and $l-t$ symbols "0". Next, we give a simple example.

Example 1: Suppose $\mathcal{S} = \{00, 01, 10, 11\}$, $l = 4$, $t = 1$. Then $\mathcal{M} = \{0001, 0010, 0100, 1000\}$. Note that $|\mathcal{M}| = \binom{4}{1} \geq |\mathcal{S}|$. Suppose further the following source encoding rule: $e_s(00) = 0001$, $e_s(01) = 0010$, $e_s(10) = 0100$ and $e_s(11) = 1000$. In order to communicate, for example, the source state "10" to a designated receiver, the transmitter emits a signal described by the following sequence: $0p00$. This sequence is interpreted as follows: only during the period of the second symbol does the sender emit some signal (energy) over the used channel. An important condition for I-codes to work properly, is that it should not be possible to block emitted signals without being detected. At the end of this Section, we discuss a possible strategy to ensure this condition.

We next discuss some properties of I-codes. Let us first introduce some additional notation. Let p_0 and p_1 denote two threshold power levels with $p_1 > p_0$. Let T_s denote the duration of an I-code symbol transmission at the physical layer. We further denote with $p_r(i)$ ($i \in \mathbb{N}$), the average power received in the interval (of duration T_s) corresponding to the ith symbol in the communicated sequence of symbols. We will assume for the moment that the users are synchronized at the physical layer (in Section III-C.3 we will discuss how this can be achieved, and address some other important practical considerations). Now, let the following be the receiver's demodulation strategy for a given symbol $i \in \mathbb{N}$: (1) if $p_r(i) \geq p_1$, output symbol "1", (2) if $p_r(i) \leq p_0$, output symbol "0", and (3) if $p_0 < p_r(i) < p_1$, output "Abort" message.

Property 1: Assume that we can ensure $p_r(i) > p_0$, for all i corresponding to symbol "1", except with a negligible probability. Then, an adversary can neither add nor remove a symbol "1" from the communicated sequence without being detected.

In order to change the sequence being transmitted, an adversary has to reposition at least one symbol "1" transmitted by a legitimate party (due to the construction of I-codes; the number of symbols "1" is fixed to t). This means that an adversary has to add a new symbol "1", as well as remove one of the old symbols "1" from the communicated sequence of symbols. However, according to the assumption in Property 1, an adversary cannot remove a symbol "1" without provoking at least the "Abort" message by the receiver. Therefore, any such attempt by the adversary will be detected.

To correctly apply I-codes, the sender has to make sure that the following condition[5] is met: *The receiver is turned on and is listening on the (correct) channel during the sender's transmission.*

2) Application of I-codes to the DH-SC protocol: The application of I-codes to the DH-SC protocol is straightforward. The resulting protocol (DH-IC) is shown on Fig. 4. The point to observe here is that we simply encode authentication value i_A and transmit it by using I-codes. The following theorem is an immediate consequence of Property 1:

Theorem 2: Assuming that the condition of Property 1 is met, DH-IC is a secure protocol enabling the authentication of the DH public parameters.

[4]Messages to be encoded by the I-code.

[5]In addition to the condition of Property 1, i.e., $p_r(i) \geq p_1$ for all i corresponding to symbol "1".

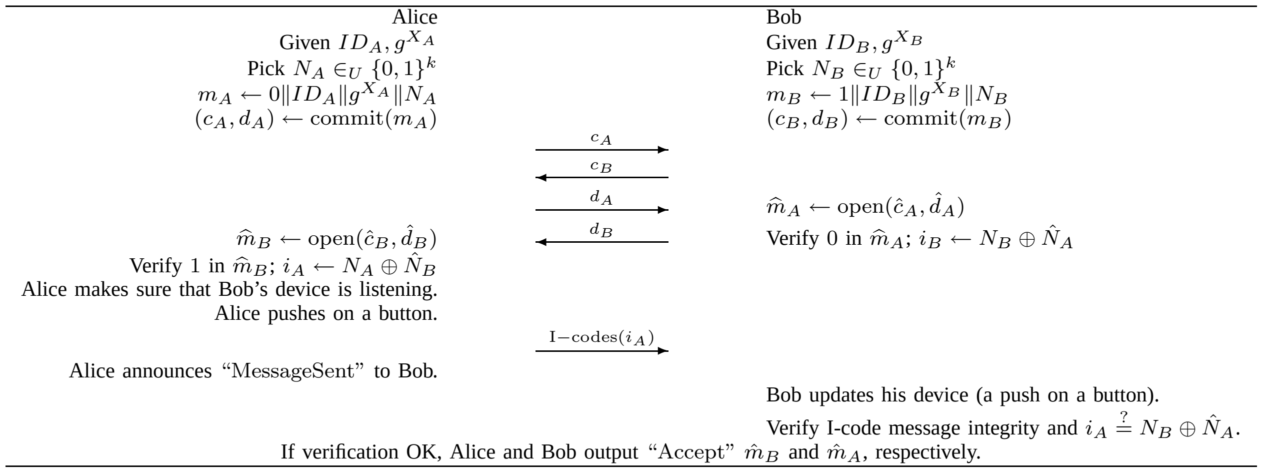
Alice
Given ID_A, g^{X_A}
Pick $N_A \in_U \{0,1\}^k$
$m_A \leftarrow 0\|ID_A\|g^{X_A}\|N_A$
$(c_A, d_A) \leftarrow \text{commit}(m_A)$
Bob
Given ID_B, g^{X_B}
Pick $N_B \in_U \{0,1\}^k$
$m_B \leftarrow 1\|ID_B\|g^{X_B}\|N_B$
$(c_B, d_B) \leftarrow \text{commit}(m_B)$
c_A
c_B
d_A
$\widehat{m}_A \leftarrow \text{open}(\hat{c}_A, \hat{d}_A)$
d_B
$\widehat{m}_B \leftarrow \text{open}(\hat{c}_B, \hat{d}_B)$
Verify 0 in $\widehat{m}_A$; $i_B \leftarrow N_B \oplus \hat{N}_A$
Verify 1 in $\widehat{m}_B$; $i_A \leftarrow N_A \oplus \hat{N}_B$
Alice makes sure that Bob's device is listening.
Alice pushes on a button.
$\text{I-codes}(i_A)$
Alice announces "MessageSent" to Bob.
Bob updates his device (a push on a button).
Verify I-code message integrity and $i_A \overset{?}{=} N_B \oplus \hat{N}_A$.
If verification OK, Alice and Bob output "Accept" $\widehat{m}_B$ and $\widehat{m}_A$, respectively.

Fig. 4. DH-SC key agreement protocol enhanced with I-codes (DH-IC protocol)

3) Some practicalities: In this section we briefly discuss the most important practical considerations around I-codes, notably in order to show their feasibility.

Security of I-codes. The security of I-codes relies on Property 1. Thus, if we can ensure that any symbol "1" cannot be completely blocked (deleted, annihilated, cancelled) once transmitted over a public channel, except with a negligible probability, then an adversary cannot change an "I-coded" message without being detected. In other words, we need to ensure that $p_r(i) > p_0$, for all i corresponding to symbol "1".

In the context of a radio channel, cancelling a radio signal in such a way that $p_r(i) \leq p_0$, where p_0 is set to the background noise level, entails sending out the inverted signal that will have exactly the same characteristics (signal level, phase) on the receiver's side. Note that $p_r(i)$ corresponds to the average power received in the interval of duration T_s. Ensuring that $p_r(i) \leq p_0$ in this setting could be quite challenging to achieve for an adversary. Indeed, the goal of the legitimate receiver is only to detect the presence of the signal at any time during the period T_s, not to completely reconstruct the signal.

To further increase the robustness of I-codes, the transmitter can have a large number of symbol "1" waveforms, one of which the transmitter chooses randomly for each symbol "1" transmitted. In this way, the attacker does not know what waveform to try to cancel. At the same time, the receiver only measures the energy he receives during intervals of duration T_s, and so any of the "1" waveforms are equally good for this purpose. Note that the receiver does not have to know which waveform the transmitter uses in a given time interval.

Power levels. For the proper decoding of I-codes, the users have to ensure that the the average power received by a designated receiver during the period T_s corresponding to a symbol "1" is greater or equal to the threshold power level p_1 (i.e., $p_r(i) > p_1$, for all i corresponding to symbol "1"). This condition can be satisfied with high probability by using relatively high transmission powers and ensuring that the distance (d) between the sender and the designated receiver is relatively short.

Synchronization. The receiver has to know when it should start demodulating the message being sent by the transmitter. This A possible solution is to use the convention that every sequence of I-code symbols begins with symbol "1". Note further that the duration of each symbol is fixed to T_s, which is a public value. For a relatively short message to be encoded by I-codes (e.g., around 50 bits with the DH-SC protocol enhanced with I-codes), we can afford the duration of an I-code symbol (T_s) to be as large as 1 to 10 milliseconds. In this case, the synchronization between the sender and the designated receiver should not be a problem. It would then take between 0.05 to 0.5 second to transfer 50 bits; which seems acceptable, given that the secrecy of the communication is at stake.

IV. Security analysis - proof of Theorem 1

In this section, we analyze the security of our key agreement protocols, primarily DH-SC protocol.

A. Overview of the approach

For the analysis of our protocols, we use the modular approach proposed by Bellare, Canetti, and Krawczyk [20]. This approach assumes two adversarial models: the authenticated link model (AM) and the un-authenticated links model (UM). The AM model is an ideal-world model in which the attacker is able to invoke protocol runs, masquerade as protocol principals, and find old session keys; however, he is not able to fabricate or replay messages that appear to come from uncorrupted parties. The UM model is a real-world model, in which the attacker can do all that it can in the AM model; in addition, he can replay messages and try to fabricate messages.

The security of the protocol is first proven in the AM model, assuming (as assumed by the model itself) that all the communication between the parties is authenticated. If the protocol is proven to be secure in the AM model, then it can be shown to be secure in the UM model, provided that each message transmitted between the parties is authenticated by a protocol called message transfer (MT) authenticator. In [20], Bellare, Canetti, and Krawczyk show that the basic Diffie-Hellman protocol is secure in the AM model, and that

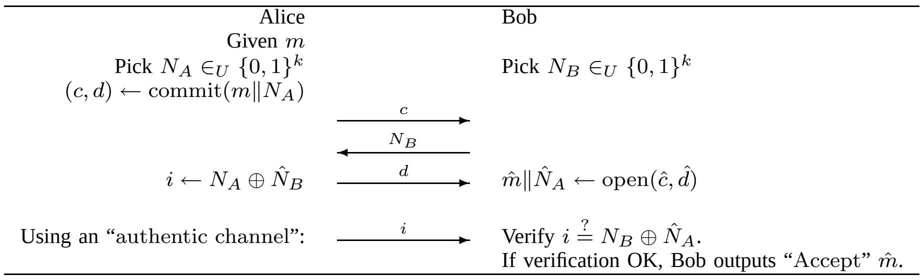
Alice
Bob
Given m
Pick $N_A \in_U \{0,1\}^k$
Pick $N_B \in_U \{0,1\}^k$
$(c,d) \leftarrow \text{commit}(m\|N_A)$
c
N_B
$i \leftarrow N_A \oplus \hat{N}_B$
d
$\hat{m}\|\hat{N}_A \leftarrow \text{open}(\hat{c},\hat{d})$
Using an "authentic channel":
i
Verify $i \overset{?}{=} N_B \oplus \hat{N}_A$.
If verification OK, Bob outputs "Accept" $\hat{m}$.

Fig. 5. Message authenticator based on short strings comparison (MTSC). Bob checks whether the source of the message $\hat{m}$ is Alice; an "authentic channel" can be implemented through visual or vocal comparison of the output strings $N_A \oplus \hat{N}_B$ and $N_B \oplus \hat{N}_A$.

it is secure in the UM model, provided that correct MT authenticators are used to authenticate transfers of DH public parameters. In their work, they use MT authenticators based on digital signatures and encryption.

Since our DH-SC, DH-DB and DH-IC protocols are all variants of the authenticated Diffie-Hellman protocol (with string-comparison-based, distance-bounding-based, and integrity-code-based authentications, respectively), showing that these protocols are secure simply requires showing that their authenticators are secure. For the DH-SC protocol, we will show the construction of the MT string-comparison-based (MTSC) authenticator and provide a detailed proof.

The MTSC authenticator is shown on Fig. 5. We analyze the security of the MTSC authenticator in the Bellare and Rogaway two-party model based on matching conversations [21], [7].

B. Matching conversations

In this model, a protocol $\Pi(k, I)$ is executed by a pair of parties $(A, B) \in I$, where I is a set of parties that share some common context (e.g., they all run a message authentication protocol). By $\Pi^t_{B,A}$ we mean that a party B attempts to authenticate a message from party A in a session that B believes has the session identifier $t \in \mathbb{N}$. Here, by authentication of a message we mean that at the end of a successful run of the protocol, party B accepts that a message m it has received must have been sent by party A, except with a negligible probability.

We consider an active attacker Mallory in the communication model of Bellare and Rogaway [21], meaning that Mallory can observe, modify and schedule communication between a pair of parties (A, B). Given that Mallory is a powerful attacker, we let Mallory interact with $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ as oracles in a "black box" style, meaning that Mallory can query $\Pi^s_{A,B}$ by supplying A with input queries that comply to the observed authentication protocol. In the response to any query, oracle $\Pi^s_{A,B}$ outputs a message that complies to the authentication protocol. We use the following format (A, B, s, conv) to record all queries and responses that $\Pi^s_{A,B}$ sent out in the session that A marks as $s \in \mathbb{N}$; we do "the same" for $\Pi^s_{B,A}$. Here, *conv* denotes a conversation of $\Pi^s_{A,B}$, meaning a sequence of timely ordered messages that $\Pi^s_{A,B}$ has sent out and received. We say that $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ have matching conversations, if for each message m sent out by $\Pi^s_{A,B}$ in time τ_i, $\Pi^t_{B,A}$ received the same message m in τ_{i+1} and if for each message m sent out by $\Pi^t_{B,A}$ in time τ_i, $\Pi^s_{A,B}$ received the same message m in τ_{i+1} [21]. Here, $\tau_0 < \tau_1 < \tau_2 < ... < \tau_R$ is, for some positive integer R, a time sequence recorded by $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ when conversing.

Consider a pair of oracles $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ that belong to party A and party B, respectively. Following the unfolding of the protocol on Fig. 5, the conversations of $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ can be written as follows:

$$\begin{aligned} \mathit{conv}_A &= (\tau_0, \perp, c),\ (\tau_2, \hat{N}_B, d),\ (\tau_4, \perp, i); \\ \mathit{conv}_B &= (\tau_1, \hat{c}, N_B),\ (\tau_3, \hat{d}, \perp),\ (\tau_5, i, \perp); \end{aligned} \quad (1)$$

where $\perp$ means that a party receives/sends no message in the corresponding time τ_i. We first observe that if the two conversations are not modified by adversary M, $\Pi^t_{B,A}$ (and hence $\Pi^t_{A,B}$) will reach the "Accept" decision and conv_A and conv_B will be matching. This is obvious because then $c = \hat{c}$, $d = \hat{d}$ (which implies $N_A = \hat{N}_A$) and $N_B = \hat{N}_B$, and therefore i_A matches $N_B \oplus \hat{N}_A$, meaning that $\Pi^t_{B,A}$ will output "Accept" (Fig. 5). Moreover, $\tau_0 < \tau_1 < \tau_2 < \tau_3 < \tau_4 < \tau_5$. This essentially means that party B will believe that the message $\hat{m}$ was sent by party A.

Definition 3: We say that $\Pi(k, (A, B))$ is a secure authentication protocol between A and B if attacker M cannot win, except with a satisfactorily small probability $\mathcal{O}(2^{-k})$ (i.e., constant c in $\mathcal{O}(\cdot)$ is such that $c \ll 2^k$). Here, M wins if $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ reach the "Accept" decision while they do not have matching conversations.

Observe that if any of $\hat{c}$, $\hat{N}_B$, $\hat{d}$ or i are missing, $\Pi^t_{A,B}$ and $\Pi^t_{B,A}$ will simply "Abort" the protocol $\Pi(k, I)$ and adversary M will certainly fail to convince $\Pi^t_{B,A}$ and $\Pi^s_{A,B}$ to "Accept".

C. Security of the MTSC-authenticator

We denote the MTSC authenticator as a protocol $\Pi(k, I)$. We observe a pair of parties $(A, B) \in I$ running $\Pi(k, I)$ and a powerful polynomially-bounded active attacker Mallory. We assume the adversary M does not belong to the set I; this is consistent with the model of Bellare and Rogaway [21] and with the fact that any two parties A and B running $\Pi(k, I)$ mutually trust each other[6].

In our security proof of $\Pi(k, I)$, we consider the $\text{commit}(\cdot)$ function to be an ideal commitment. We further assume that each party has access to a perfect random number generator.

[6]Otherwise, what is the point of receiving message m from a dishonest party?

Note that we will observe the security of $\Pi(k, I)$ in the sense of Definition 3. Let γ be the maximum number of sessions (successful or abortive) that any party can participate in. We will assume that there are at most n parties using protocol $\Pi(k, I)$. In our analysis, we will also assume that each party participates in at most one message authentication session at a time.

We show that for fixed A, B and t, the probability that oracle $\Pi^t_{B,A}$ outputs "Accept" without a matching conversation is satisfactorily small. Note that if $\Pi^t_{B,A}$ outputs "Accept" then there must exist some oracle $\Pi^s_{A,B}$ (with party A) that outputs "Accept" too; message i, at the end of protocol $\Pi(k, I)$, guarantees this. We first state the following intuitive result:

Lemma 1: If adversary M is to succeed against a pair of oracles $(\Pi^s_{A,B}, \Pi^t_{B,A})$, then we must have $c \neq \hat{c}$, where c is the commitment sent out by $\Pi^s_{A,B}$ and $\hat{c}$ is the commitment received by $\Pi^t_{B,A}$.

Proof: Claim: If $c = \hat{c}$ and $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ both "Accept", then $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ must have matching conversations. Indeed, M cannot break the used ideal commitment scheme, so we must have $d = \hat{d}$ and hence $m = \hat{m}$ and $N_A = \hat{N}_A$. Furthermore, since $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ both "Accept", we have $N_A \oplus \hat{N}_B = N_B \oplus \hat{N}_A$ and hence $N_B = \hat{N}_B$. Moreover, $\tau_0 < \tau_1 < \tau_2 < \tau_3 < \tau_4 < \tau_5$. Therefore, $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ have matching conversations. ■

If $\Pi^t_{B,A}$ is to output "Accept", then the pair $(\hat{c}, \hat{d})$ has to be a valid commit/opening pair. This is because M cannot break the used ideal commitment scheme; any attacking attempt that involves breaking such a commitment scheme will fail with probability 1. Furthermore, if oracle $\Pi^t_{B,A}$ is to output "Accept", then there must exist some $\Pi^s_{A,B}$ (with party A) that outputs $i = N_A \oplus \hat{N}_B$ such that $N_A \oplus \hat{N}_B = N_B \oplus \hat{N}_A$. Note here that $\hat{N}_A$ and $\hat{N}_B$ are potentially chosen by the adversary M.

Consider now the interaction between a pair of oracles $(\Pi^s_{A,B}, \Pi^t_{B,A})$ and adversary M as given in (1). Assume that $(\hat{c}, \hat{d})$ is a valid commit/opening pair (i.e., M does not try to break the commitment scheme) and assume $c \neq \hat{c}$ (Lemma 1). Note that if any of the two assumptions does not hold, then M certainly fails. Then, we have the following:

Lemma 2: For any such interaction between $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ and adversary M, we have $Pr[N_A \oplus \hat{N}_B = N_B \oplus \hat{N}_A] \leq 2^{-k}$.

Proof: Observe that M has to submit $\hat{N}_B$ before actually seeing N_A. This follows from the unfolding of $\Pi(k, I)$ and the hiding property of the commitment scheme. Similarly, M has to submit $\hat{N}_A$ (as a part of commitment $\hat{c}$) before actually seeing N_B. This follows from the unfolding of $\Pi(k, I)$ and the binding property of the commitment scheme. Thus, irrespectively of the attacking strategy taken by M, one among N_A and N_B will certainly be disclosed as the last value over an unauthentic channel. If it happens that both N_A and N_B are disclosed at the same time, then we pick an arbitrary one.

Assume that N_A is disclosed as the closing value. Then, we have $Pr[N_A \oplus \hat{N}_B = N_B \oplus \hat{N}_A] = Pr[N_A = N_B \oplus \hat{N}_A \oplus \hat{N}_B] \leq 2^{-k}$, that is, N_A and N_B are independent and uniformly distributed random variables, and $\hat{N}_A$ and $\hat{N}_B$ are both generated before N_A. The same holds for N_B as the closing value. Therefore, $Pr[N_A \oplus \hat{N}_B = N_B \oplus \hat{N}_A] \leq 2^{-k}$. Note that the assumption $c \neq \hat{c}$ precludes from trivial situations, where M would not modify the messages, to take place; in which case we would have $Pr[N_A \oplus \hat{N}_B = N_B \oplus \hat{N}_A] = 1$. ■

From Lemma 2, we conclude that the probability that there exists oracle $\Pi^t_{B,A}$ that belongs to party B and that "Accepts" without a matching conversation is at most 2^{-k} times the maximum number of interactions (successful or abortive) that party B has participated in. It is crucial that we take abortive attempts into account, too, when evaluating the probability that M is successful against a given party. This is because M learns that his attempt is unsuccessful (i.e., $N_A \oplus \hat{N}_B \neq N_B \oplus \hat{N}_A$) before M potentially sends out $\hat{d}$ in an attempt to disclose $\hat{N}_A$ to party B. If M is not successful in a given attempt, he can simply abort the protocol by simply not sending $\hat{d}$ to B.

Since we limit each party to participate in at most γ successful or abortive runs of $\Pi(k, I)$, the probability that there exists oracle $\Pi^t_{B,A}$ that belongs to party B and that "Accepts" without a matching conversation is at most $\gamma 2^{-k}$. Note that party A "Accepts" only if the corresponding party B "Accepts". Therefore, the probability that there exists oracle $\Pi^s_{A,B}$ that belongs to party A and that "Accepts" without a matching conversation is at most $\gamma 2^{-k}$. Finally, the probability that any party is broken, assuming that there are n parties that use protocol $\Pi(k, I)$, is at most $n\gamma 2^{-k}$.

D. From secure MT-authenticator to secure DH-SC protocol

In the previous section, we have proved that MTSC emulates the message transmission (MT) protocol in unauthenticated networks (in the real world), meaning that if party B (and hence, party A) "Accept" at the end of the protocol shown on Fig. 5, then B knows that the message he received was sent by A (except with a probability $n\gamma 2^{-k}$; for appropriately chosen k, $n\gamma 2^{-k}$ can be made satisfactorily small).

It is proven in [20] that the basic DH protocol is secure in the AM model, assuming that the Decisional DH problem is intractable. Now, to obtain a secure DH protocol in unauthenticated networks (UM model), we simply apply the MTSC-authenticator to each message of the basic DH protocol. By doing this we obtain a DH protocol (DH-SC) that is secure in the UM model (Theorem 3 in [20]). A naive application of the MTSC-authenticator (Fig. 5) to the basic DH protocol results in a protocol that involves 6-messages and 2 string comparisons. A simple way to improve on this is to piggyback the messages of one MTSC-authenticator on the other as in the DH-SC protocol (Fig. 1): in this case, the k-bit random string N_A (sent through the commit/opening pair (c_A, d_A)) plays two roles: (1) the role of N_A in the MTSC-authenticator on Fig. 5, and (2) the role of N_B in the MTSC-authenticator on the same figure (in this case, Alice from Fig. 1 wants to make sure that a message she received was sent by Bob). The first role is clearly fulfilled. That N_A on Fig. 1 also fulfills the second role follows from the fact that N_A on Fig. 1 remains hidden until Alice opens c_A by sending out d_A and Alice sends out d_A only after receiving $\hat{c}_B$. The same analysis is valid for Bob from Fig. 1.

It is important to note that with the above optimization of both MT-message authenticators, we introduce a vulnerability to the *reflection attack* [7] in the DH-SC protocol (Fig. 1). For example, M can simply reflect all the messages he receives back to the messages' originators. Note that in this case $N_A \oplus N_A = N_B \oplus N_B$. This is why we append two public (fixed) values 0 and 1 to messages m_A and m_B, respectively, and impose the verification of both at the end of the protocol on Fig. 1. Note that after having thwarted the reflection attack, the DH-SC protocol can be seen as two runs of the MTSC-authenticator (Fig. 5). With this, we conclude the proof of Theorem 1.

V. Related work

The problem of key establishment is a very active area of research. Stajano and Anderson propose the *resurrecting duckling* security policy model, [22] and [1], in which key establishment is based on the physical contact between communicating parties (their PDAs). A physical contact acts as a *location limited channel*, which can be used to transmit a key (or a secret) in plaintext. Thus, no cryptography is required at this stage[7]. The potential drawback of this approach is that the realization of a physical contact can be cumbersome with bulky devices (e.g., laptops).

An approach inspired by the resurrecting duckling security policy model is proposed by Balfanz et al. [2]. In this work, the authors go one step further and relax the requirement that the location limited channel has to be secure against passive eavesdropping; they introduce the notion of a *location-limited channel* (e.g., an infrared link). A location-limited channel is used to exchange pre-authentication data and should be resistant to active attacks (e.g., man-in-the-middle). Once pre-authentication data are exchanged over a location-limited channel, users switch to a common radio channel and run any standard key exchange protocol over it. Possible candidates for a location-limited channel include: physical contact, infrared, and sound (ultrasound) [2]. Here again, the disadvantage of this approach is that it may be a cumbersome to realize a link with bulky devices (e.g., laptops) in the case of infrared or physical contact. In addition, the infrared link itself is not well studied in the context of secure communications. Actually, our DH-SC protocol could be applied to the infrared link as well.

Asokan and Ginzboorg propose another solution based on a shared password [23]. They consider the problem of setting up a session key between a group of people (i.e., their computers) who get together in a meeting room and who share no prior context. It is assumed that they do not have access to public key infrastructure or third party key management services. The proposed solution is the following. A fresh password is chosen and shared among those present in the room (e.g., by writing it on a sheet of paper or a blackboard). The shared password is then used to derive a strong shared session key. This approach requires users to type the chosen password into their personal devices.

[7]This means that the location limited channel should be resistant to eavesdropping, a reasonable assumption in this case.

It is well known that IT security systems are only as secure as their weakest link. In most IT systems the weakest links are the users themselves. People are slow and unreliable when dealing with meaningless strings, and they have difficulties remembering strong passwords. In [24], Perrig and Song suggest using hash visualization to improve the security of such systems. Hash visualization is a technique that replaces meaningless strings with structured images. However, having to compare complex images can be cumbersome.

In US patent no. 5,450,493 [15], Maher presents several methods to verify DH public parameters exchanged between users. The first method described in [15] is the most relevant one for the problem we consider in this paper; other methods are based on certificates and/or shared secrets. Thus, A and B first perform the DH key exchange protocol and in turn report to each other values $f(K_A)$ and $f(K_B)$, where K_A and K_B are the shared DH keys as computed by A and B, respectively, and f is a compression function (i.e., f maps a key to 4-digit hex vectors [15]). Unfortunately, this technique has a flaw, which was discovered by Jakobsson [25]. The problem with Maher's technique is the following. An attacker Mallory M, who knows f and controls all the communication, first generates his secret exponents X_1 and X_2 and the corresponding public parameters g^{X_1} and g^{X_2}. Since M knows that A and B will compare $f(g^{X_A X_2})$ and $f(g^{X_B X_1})$, he checks if $f(g^{X_A X_2}) = f(g^{X_B X_1})$. If this is the case, M sends g^{X_2} instead of g^{X_B} to A, and g^{X_1} instead of g^{X_A} to B. If $f(g^{X_A X_2}) \neq f(g^{X_B X_1})$, M generates new values for X_1 and X_2 and repeats the above procedure. Since f outputs a very short string (4-digit hex vector [15]), M will find a collision after a relatively low number of attempts.

Motivated by the flaw in [15], Jakobsson [25] and Larsson [26] proposed two solutions. However, both solutions are based on a temporary secret shared between the two users (thus, for example, SHAKE stands for *Shared key Authenticated Key Exchange*). In our paper, we consider the same problem but in a more demanding setting, as we assume that the users share no secret key prior to the key exchange.

Dohrmann and Ellison [27] propose a method for key verification that is similar to our approach; this method is based on converting key hashes to readable words or to an appropriate graphical representation. However, it seems that users are required to compare a substantial number of words (or graphical objects); this task could take them as much as 24 seconds according to [27]. This time is significantly reduced when the graphical representation is used. However, Dohrmann and Ellison provide no security analysis of their approach.

In [3] and [28], Gehrmann et. al., propose a set of techniques to enable wireless devices to authenticate one another via an insecure wireless channel with the aid of the manual transfer of data between the devices. The protocol, which they call MANA II, is similar to our DH-SC protocol; in both protocols the parties have to compare the output of their devices. The MANA II protocol is based on authentication codes. At the end of the protocol the parties have to compare a key and a check value, where only the check value contributes to the uncertainty of the attacker. As a result, with MANA II the number of bits to be compared by the parties is twice as much as with our DH-SC. Other mechanisms proposed by [3] and [28] basically require the users to type in given values into their devices. The important difference between MANA II and our DH-SC protocol is that MANA II requires the parties to compare two strings (a key and a check value), whereas only one string (the check value) contributes to the uncertainty of the attacker. As a result, for a fixed security level of, MANA II requires the parties to compare twice as many bits as in the case of the DH-SC protocol.

We should mention other key-exchange protocols, proposed primarily for the use in the Internet: IKE [29], JFK [30] and SIGMA [31]. All these protocols involve authentication by means of digital signatures, which clearly does not fit the problem we study here. We also should mention the work of Corner and Noble [32], who consider the problem of transient authentication between a user and his device, as well as the work of Čapkun et. al [33], where the authors show how to make use of users mobility to bootstrap secure communication in open ad hoc networks.

VI. Conclusion

In this paper, we have provided three solutions to the fundamental problem of key agreement over a radio link. As user-friendliness is extremely important for the acceptance of any security scheme, we have minimized the burden on the user: there is no need of physical contact, nor of infrared communication between the devices.

To our best knowledge, the two last proposals (based on distance bounding and on I-codes) are new. Likewise, the security analysis of the first proposal (based on the comparison of short strings) and the MT-authenticator based on short string verification are also new.

Acknowledgements

The authors would like to thank Imad Aad, Levente Buttyan, Markus Jakobsson, Emre Telatar and Serge Vaudenay for helpful discussions and comments. We also thank anonymous reviewers for their valuable input.

References

[1] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks," in *Proc. 7th Int. Workshop on Security Protocols*, Cambridge, UK, 1999, pp. 172–194.

[2] D. Balfanz, D. Smetters, P. Stewart, and H. Wong, "Talking to Strangers: Authentication in Ad Hoc Wireless Networks," in *Proc. 9th Annu. Network and Distributed System Security Symp.*, San Diego, California, USA, 2002. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/02/proceedings/papers/balfan.pdf

[3] C. Gehrmann, C. Mitchell, and K. Nyberg, "Manual Authentication for Wireless Devices," RSA Cryptobytes, Vol. 7, No. 1, pp. 29–37, 2004.

[4] J.-H. Hoepman, "The Ephemeral Pairing Problem," in *Financial Cryptography*, ser. LNCS, vol. 3110, Key West, Florida, USA, February 2004, pp. 212–226.

[5] M. Čagalj and J.-P. Hubaux, "Key agreement over a radio link," EPFL-IC-ICA, Tech. Rep. IC/2004/16, January 2004.

[6] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. Inform. Theory*, vol. 22, no. 6, pp. 644–654, November 1976.

[7] W. Mao, *Modern Cryptography, Theory & Practice*. Prentice Hall PTR, 2004.

[8] D. Plummer, "An Ethernet Address Resolution Protocol," IETF Standard, RFC 826, 1982.

[9] D. Song, "dsniff." [Online]. Available: http://naughty.monkey.org/∼dugsong/dsniff/

[10] "Specification of the Bluetooth System (Core). Version 1.1," 2001. [Online]. Available: http://www.bluetooth.org

[11] M. Jakobsson and S. Wetzel, "Security Weaknesses in Bluetooth," in *CT-RSA*, ser. LNCS, vol. 2020, San Francisco, CA, USA, February 2001, pp. 176–191.

[12] D. Kügler, "Man in the Middle Attacks on Bluetooth," in *Financial Cryptography*, ser. LNCS, vol. 2742, Guadeloupe, French West Indies, January 2003, pp. 149–161.

[13] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11-1999*, The Institute of Electrical and Electronics Engineers, New York Std.

[14] N. Haller, C. Metz, P. Nesser, and M. Straw, "A One-Time Password System," RFC 2289, February 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2289.txt?number=2289

[15] D. Maher, "Secure communication method and apparatus," U.S. Patent 5,450,493, December 29, 1993.

[16] S. Brands and D. Chaum, "Distance-bounding protocols," in *EUROCRYPT*, ser. LNCS, vol. 765, Lofthus, Norway, 1993, pp. 344–359.

[17] R. Fontana, "Experimental Results from an Ultra Wideband Precision Geolocation System," *Ultra-Wideband, Short-Pulse Electromagnetics 5*, pp. 215–224, January 2002.

[18] N. Sastry, U. Shankar, and D. Wagner, "Secure Verification of Location claims," in *Proc. ACM Workshop on Wireless Security (WiSe)*, September 2003, pp. 1–10.

[19] B. Waters and E. Felten, "Secure, Private Proofs of Location," Princeton University, Tech. Rep. TR-667-03.

[20] M. Bellare, R. Canetti, and H. Krawczyk, "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols," in *Proc. 30th Annu. Symp. Theory of Computing*, Dallas, Texas, United States, 1998, pp. 419 – 428.

[21] M. Bellare and P. Rogaway, "Entity Authentication and Key Distribution," in *Proc. 13th Annu. Int. Cryptology Conference*, Santa Barbara, California, USA, 1993, pp. 232–249.

[22] F. Stajano, *Security for Ubiquitous Computing*. John Wiley & Sons, Ltd., 2002.

[23] N. Asokan and P. Ginzboorg, "Key Agreement in Ad-hoc Networks," *Computer Communications*, vol. 23, no. 17, pp. 1627–1637, 2000.

[24] A. Perrig and D. Song, "Hash Visualization: A New Technique to Improve Real-World Security," in *Proc. 1999 Int. Workshop on Cryptographic Techniques and E-Commerce*, 1999, pp. 131–138.

[25] M. Jakobsson, "Issues in Security and Privacy (lecture slides)." [Online]. Available: http://www.informatics.indiana.edu/markus/i400/

[26] J.-O. Larsson and M. Jakobsson, "SHAKE," Private communication with M. Jakobsson.

[27] C. Ellison and S. Dohrmann, "Public-key support for group collaboration," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 4, pp. 547–565, 2003.

[28] C. Gehrmann and K. Nyberg, "Enhancements to Bluetooth Baseband Security," in *Proc. 6th Nordic Workshop on Secure IT Systems*, Copenhagen, Denmark, November 2001, pp. 39–53.

[29] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," RFC 2409, November 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2409.txt?number=2409

[30] W. Aiello, S. M. Bellovin, M. Blaze, R. Canettia, J. Ioannidis, A. D. Keromytis, and O. Reingold, "Efficient, DoS-Resistant, Secure Key Exchange for Internet Protocols," in *Proc. ACM Computer and Communications Security Conference*, Washington, DC, USA, 2000, pp. 48–58.

[31] H. Krawczyk, "The SIGMA Family of Key-Exchange Protocols." [Online]. Available: http://www.ee.technion.ac.il/∼hugo/sigma.html

[32] M. Corner and B. Noble, "Protecting File Systems with Transient Authentication," *Wireless Networks*, vol. 11, no. 1-2, pp. 7–19, January 2005.

[33] S. Čapkun, J.-P. Hubaux, and L. Buttyan, "Mobility Helps Peer-to-Peer Security," IEEE Trans. Mobile Comput., 2004, to be published.